(4)

ETL-0495

Parallel algorithms for computer vision, second year report

Tomaso Poggio
James Little

Massachusetts Institute of Technology
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

March 1988

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

U.S. Army Corps of Engineers
Engineer Topographic Laboratories
Fort Belvoir, Virginia 22060-5546

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A202 [illegible]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) ETL-0495
6a. NAME OF PERFORMING ORGANIZATION Massachusetts Institute of Technology; 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION U.S. Army Engineer Topographic Laboratories
6c. ADDRESS *(City, State, and ZIP Code)* Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139	7b. ADDRESS *(City, State, and ZIP Code)* Fort Belvoir, VA 22060-5546
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency; 8b. OFFICE SYMBOL *(If applicable)* ISTO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DACA76-85-C-0010
8c. ADDRESS *(City, State, and ZIP Code)* 1400 Wilson Boulevard, Arlington, VA 22209-2308	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Parallel Algorithms for Computer Vision Second Year Report

12. PERSONAL AUTHOR(S)
Tomaso Poggio and James Little

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Annual	FROM 8/31/86 TO 8/31/87	1988 March	22

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Computer Vision
			Parallel Algorithms & Architectures

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Much ~~of our~~ the work during the past year has focused on building our Vision Machine system. The Vision Machine is a testbed for ~~our~~ research on parallel vision algorithms and their integration. The system consists of an input device--a movable two camera Eye-Head system with six degrees of freedom--and the 16K Connection Machine (CM-1). We have concentrated on implementing and testing early vision algorithms, and on developing a new sophisticated techniques for their integration. The output of the integration stage will be used for navigation and recognition tasks.

From August 31, 1986 to August 31, 1987, we have been using the Connection Machine delivered on July 31, 1986 by Thinking Machines Corporation ~~(TMC)~~. We have developed and tested a substantial body of vision software on the machine. We have also nearly completed, well ahead of schedule, the development of an integrated Vision Machine that includes several early vision algorithms, and the integration stage of middle vision. As outlined in our original proposal, we have begun to explore parallel algorithms at the higher level of recognition. We have also studied the performance of alternative, nonconventional architectures for navigation, and worked on the difficult issue of alternative parallel languages for the Connection Machine, in addition to *LISP and C*. The body of this report gives an overview of the results of our research during the second twelve month of funding. Details can be found in the attached publications.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Rosalene M. Holecheck	22b. TELEPHONE *(Include Area Code)* (202) 355-2700; 22c. OFFICE SYMBOL CEETL-RI

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

[illegible]

PARALLEL ALGORITHMS FOR COMPUTER VISION

U.S. Army Topographic Laboratories
Contract Number DACA76-85-C0010

Tomaso Poggio and James Little
August 31, 1986—August 31, 1987

Second Year Report

TABLE OF CONTENTS

1. SUMMARY

1.1. The Vision Machine

Much of our work during the past year has focused on building our Vision Machine system. The Vision Machine is a testbed for our research on parallel vision algorithms and their integration. The system consists of an input device – a movable two camera Eye-Head system with six degrees of freedom - and the 16K Connection Machine (CM-1). We have concentrated on implementing and testing early vision algorithms, and on developing a new sophisticated technique for their integration. The output of the integration stage will be used for navigation and recognition tasks.

1.2. An Overview

From August 31, 1986 to August 31, 1987, we have been using the Connection Machine delivered on July 31, 1986 by Thinking Machines Corporation (TMC). We have developed and tested a substantial body of vision software on the machine. We have also nearly completed, well ahead of schedule, the development of an integrated Vision Machine that includes several *early vision* algorithms, and the integration stage of *middle vision*. As outlined in our original proposal, we have begun to explore parallel algorithms at the higher level of recognition. We have also studied the performance of alternative, nonconventional architectures for navigation, and worked on the difficult issue of alternative parallel languages for the Connection Machine, in addition to *LISP and C*. The body of this report gives an overview of the results of our research during the second twelve months of funding. Details can be found in the attached publications.

2. ACHIEVEMENTS IN THE SECOND YEAR

We give below a brief overview of our main achievements.

2.1. The Vision Machine

The Vision Machine hardware consists of a "Eye-Head" system and a Connection Machine (CM-1). The MIT Eye-Head system is a movable robot that consists of two CCD cameras, two sets of two mirrors each, and a "head" that can be moved with two degrees of freedom by step-motors. Both the "head" and the mirrors (driven by galvanometers) are computer controlled. The system can effectively look around the ninth floor of the MIT Artificial Intelligence Laboratory building at Technology Square, and grab stereo images. In the very near future, we will add zoom, auto-focusing and color capabilities to this input device.

The Connection Machine [Hillis, 1985] is a fine-grained parallel computing machine having up to 64K processors, and operating under a single instruction stream broadcast to all processors. The MIT Connection Machine is a 16K processor CM-1. Each processor is a 1-bit serial processor, with 4K bits of memory. There are two modes of communication among the processors. In the first mode, the processors can be thought of as connected by a mesh of wires into an $m \times m$ grid network (the NEWS network, so-named for the four cardinal directions), allowing rapid direct communication between neighboring processors; m is variable. The second mode, the *router*, allows messages to be sent from any processor to any other processor in the machine. The processors in the Connection Machine can be envisioned as the vertices of a 16-dimensional hypercube. Each processor in the Connection Machine is identified by a unique integer, its hypercube address. Messages pass along the edges of the hypercube from source processors to destination processors. Besides local operations in the processors, the Connection Machine can return to the host machine the result of various operations on a field in all processors; it can return the global maximum, minimum, sum, logical AND, and logical OR of the field.

The core of the Vision Machine system is software on the CM-1 written mainly in *LISP and PARIS, which are additions to Common-Lisp that provide access to the Connection Machine. Since the Connection Machine is a Single-Instruction Multiple-Data (SIMD) machine, there is an instruction stream broadcast to the Connection Machine from the host, in our case a Symbolics Lisp Machine. The form of programming suitable for the Connection

Machine, due to its SIMD nature, is "data-level parallelism", which focuses on the movement and interaction of data. Each processor in the Connection Machine is associated with a data element. All selected data elements in a SIMD machine are manipulated in the same fashion. Because of these features of the Connection Machine, it is natural to map pixels in the images we analyze to processors, the data elements in the Connection Machine.

The Connection Machine architecture provides some useful primitive operations for early, middle and high-level vision. It is distinguished from mesh computers by its capability for general non-local communication among the processors, and from pyramid machines by its uniform bandwidth communication among all processors.

Our integration effort has to this point concentrated on integrating data produced by individual modules into a coherent description of the scene, specifically at boundaries. The memory limitations of our CM-1 force us to run each module serially, since it is not possible to represent simultaneously in the CM-1 all of the many structures involved in the texture, color, motion and stereo modules. With a CM-2, we should be able to couple these modules more tightly.

2.2. Early Vision Algorithms

2.2.1. Edge Detection

Edge detection is an important first step in many low-level vision algorithms. It generates a concise, compact description of the structure of the image, suitable for manipulation in higher-level interpretation tasks. We have implemented on the Connection Machine both the Marr-Hildreth and the Canny edge detection schemes.

A fundamental operation in vision processing is filtering the input image to remove noise and to select an appropriate spatial scale. Typically, filtering is accomplished by convolution with a filter of bounded spatial extent, often a Gaussian. We have implemented a variety of methods for computing the Gaussian convolution of an image. Primarily, we use the binomial approximation to the one-dimensional Gaussian implemented by repeated summing, using a kernel of $1/4, 1/2, 1/4$. To smooth the image with a filter approximating a Gaussian with standard deviation σ requires $2\sigma^2 - 1$ convolutions with this kernel. Each convolution needs two additions and two NEWS accesses. This method can also be used to generate an approximation to convolution with a two-dimensional Gaussian. Convolution with a Gaussian filter can also be approximated by iterated convolution with a uniform (boxcar) filter of width N and height $1/N$. To approximate a Gaussian with standard deviation σ, $12\sigma^2 - 1/N^2 - 1$ iterations are required. This approximation is useful when σ is large. We compute $\nabla^2 * G(\sigma)$ by convolving with $G(\sigma)$, retaining extra precision, and then filtering with the discrete Laplacian.

To detect zero-crossings in the Marr-Hildreth method, each processor need only examine the sign bits of neighboring processors, using NEWS accesses.

Canny Edge Detection

The Canny edge detector is based on directional derivatives, so it has improved localization. Implementing the Canny edge detector on the Connection Machine involves implementing Gaussian filtering, computing directional derivatives, non-maximum suppression, and thresholding with hysteresis. The algorithm has been implemented on the Connection Machine, and is routinely used in real-time processing tasks as an integral part of the Vision Machine system.

Gaussian filtering and computing directional derivatives are local operations as described above. In non-maximum suppression, gradient magnitudes are interpolated along lines in gradient directions. Then those pixels are selected for which the gradient magnitude is a local maximum. Thresholding with hysteresis estimates the distribution of gradient magnitudes in the image, and chooses a low threshold, *low*, below which all points are considered noise. Then a high threshold, ***high***, is computed; pixels with gradient magnitudes above ***high*** are automatically marked as edge elements. Any pixels above *low* threshold but below ***high*** become edge elements only if they can be connected to a selected pixel above ***high*** by a set of selected pixels (local maxima), above *low*. This requires propagating information along curves.

Histograms

Estimating the gradient magnitude distribution is performed by computing its histogram. There are several ways this can be implemented on the Connection Machine. Gradient magnitudes can be quantized for the histogram bucket size. Sorting these values reconfigures the data: $\ldots k, k, k, k, k+1, k+1, k+1 \ldots$. Each processor determines whether its left neighbor is less than itself. Each processor for which this holds sends its cube address to location

H_k in the histogram table, resulting in a cumulative frequency distribution table, easily converted to a histogram. Computing a histogram in m buckets by ***counting*** involves stepping from $0 \leq i \leq m$, selecting processors where intensity $= i$, and counting the number of selected processors (H_i), using m global counting operations. When m is less than 64, this can be more efficient than sorting. Finally, when only one value in the distribution is needed, for example, finding the k^{th} percentile, the percentile can be found in a binary search, using $O(\log m)$ global counting operations. Computing a Hough Transform is similar to computing a histogram. In this, we will be able to take advantage of ***a priori*** information on the distribution of values to devise a fast algorithm.

Propagation

In thresholding with hysteresis, the existence of a ***high*** value on a curve is propagated along the curve, to enable any *low* pixels to become *edge* pixels. Only pixels above *low* which survive non-maximum suppression are considered. Each pixel can, in constant time, find the neighboring pixels with which it forms a connected line, using the NEWS network. All pixels above ***high*** are marked as *edge* pixels. Currently, the program iterates, in each step marking as *edge* pixels any *low* pixels adjacent to *edge* pixels, using NEWS connections. When no pixels change state, the iteration terminates, taking a number of steps proportional to the length m of the longest chain of *low* pixels which eventually become *edge* pixels. Using ***doubling***, propagating the *edge* property changes this to $O(\log m)$. In practice, propagation in the NEWS network is faster than using the asymptotically optimal ***doubling*** procedure.

2.2.2. Stereo

Binocular stereo is one of the most precise sources of depth information. For several months we have been using a parallel implementation of a new stereo algorithm on the Connection machine. This stereo algorithm is related to schemes proposed by Marroquin [1983], Prazdny [1985], and especially Marr and Poggio [1976]. These algorithms are feature based. We have also recently explored an algorithm based on matching intensities directly. It is conceivable that a similar scheme may complement a feature based algorithms to provide a denser depth map.

Parallel Stereo and the Forbidden Zone Constraint

A new simple but fast algorithm has been implemented in collaboration with Thinking Machines Corporation on the Connection Machine [Drumheller and Poggio, 1986]. In the past months it has been reimplemented and improved by W. Gillett as an integral part of the Vision Machine system. Its features include: a) the potential for combining different primitives, including color information; b) the use of a new and stronger formulation of the uniqueness constraint; and c) a disparity representation that maps efficiently into the CM architecture. The algorithm consists of the following steps:

Compute features for matching

Compute potential matches

Determine the amount of local support for each potential match

Choose correct matches on the basis of local support and constraints on uniqueness and ordering.

The algorithm does not require a particular type of matching feature. Different types of features can be used, such as the sign of the convolution with a difference of Gaussians. Let us assume that the images are perfectly registered and that all epipolar lines are horizontal. We can represent the set of potential matches in a *one-dimensional* stereo matching problem (i.e., for a pair of epipolar lines) by the same diagram used by Marr and Poggio [1976]. This representation of the stereo problem is the starting point for mapping a stereo algorithm into the CM.

Potential matches are allowed to occur between two zero-crossings of the same sign. This implements the *compatibility constraint* [Marr and Poggio, 1976]. The first step toward distinguishing the correct matches from the false ones is to apply the *continuity constraint* [Marr and Poggio, 1976]. This principle states that since most surfaces in the real world are piecewise smooth, potential matches should be selected that result in a piecewise smooth disparity function. A straightforward way to measure how well each disparity satisfies the smoothness condition is to convolve the three-dimensional region of x-y-d—space contained by the field P with a three-dimensional kernel that gathers support from smooth configurations of potential matches. There are many different kernels, or *support functions*, which will do a good job on this task. Marr and Poggio [1976] use a very simple support function (or "excitatory region") that is circular, uniformly weighted and flat, i.e., it occupies only one level in the disparity dimension. Every potential match is surrounded by an hourglass-shaped *forbidden zone*. Within the forbidden zone there must be

no more than one match unless the scene contains transparent or narrowly-occluding objects. Examples of such special scenes include a pane of glass with markings on both sides, or a vertical wire suspended in front of a textured wall. These situations violate the ***ordering constraint*** [Yuille and Poggio, 1984]. If we assume that the scene contains only opaque objects with no narrow occlusions, then it makes sense to enforce uniqueness not only along lines of sight, but along ***any line of sight in the forbidden zone.*** A correct match should be the only match within its entire forbidden zone. The Marr-Poggio algorithm and the winner-take-all algorithms mentioned earlier [Prazdny, 1985; Pollard, Mayhew and Frisby, 1985; Marroquin, 1983] use only the left and right eye lines of sight, which comprise a small subregion of the entire forbidden zone. Notice that the forbidden zone property is reflexive: if a match lies within the forbidden zone of another, the latter is in the forbidden zone of the first [see Yuille and Poggio, 1985, in which transitivity is also proven].

The algorithm enforces uniqueness by suppressing all matches that lie within the forbidden zone of the match that gathers maximum support from the 3-D convolution operation. The process of non-maximum suppression along lines of sight is called the *winner-take-all* approach. We have used a stronger, more general version of the winner-take-all method. Instead of applying non-maximum suppression along only the left and right eye lines of sight, we apply it across the entire forbidden zone. In general, the use of the entire forbidden zone in the winner-take-all step results in fewer matches than when using just the left and right eye lines of sight. However, the number of errors almost always decreases more than the number of matches, especially in the occluded region. Therefore, the ratio of errors to matches decreases. This supports the hypothesis that the entire forbidden zone could be exploited to advantage for scenes known to contain only opaque objects with no narrow occlusions.

W. Gillett has recently implemented a very simple statistical analysis of the voting process (how close the winners are, and where ties occur) to obtain some initial information about depth discontinuities during stereo matching, rather than afterwards as in the MRF scheme. Preliminary experiments indicate that the method is feasible.

The stereo algorithm runs on the Connection Machine system with good results on natural scenes in times that are typically on the order of one second.

2.2.3. Motion

A Parallel Constraint Algorithm for Optical Flow

J. Little, H. Bülthoff and T. Poggio have developed a new, fast parallel algorithm for computing optical flow. The algorithm is based on a new regularization method that we call the "constraint method". This method, based on a theorem of Tikhonov, can enforce local constraints and lead directly to efficient, parallel algorithms. The specific constraint exploited by our algorithm can be shown to correspond, in its most general form, to 3-D rigid motion of planar surfaces. Segmentation of the motion field can be obtained from the optical flow field generated by the algorithm. An iterative scheme provides fast approximate solutions and subsequently refines them. The algorithm has been implemented on the CM, and is routinely used in real-time processing tasks as an integral part of the Vision Machine system.

The algorithm generates an optical flow field, a vector field which approximates the projected motion field. The procedure produces sparse or dense output, depending on whether edge features or intensities are used. The algorithm assumes that image displacements are small, within a range $(\pm\delta, \pm\delta)$. In addition, it is assumed that the optical flow is locally constant in a small region surrounding a point. This assumption is strictly only true for translational motion of 3-D planar surface patches parallel to the image plane. It is a restrictive assumption which, however, may be a satisfactory *local* approximation in many cases. Let $E_t(x,y)$ and $E_{t+\Delta t}(x,y)$ represent transformations of two discrete images separated by time interval Δt, such as filtered images or a map of the intensity changes in the two images (more generally, they can be maps containing a feature vector at each location (x,y) in the image).

We look for a discrete motion displacement $\underline{v} = (v_x, v_y)$ at each location x, y in the image such that

$$\|E_t(x,y) - E_{t+\Delta t}(x + v_x\Delta t, y + v_y\Delta t)\|_{\mathrm{patch}_i} = \min \tag{1}$$

where the norm is a summation over a local neighborhood centered at each location (x,y); $\underline{v}(x,y)$ is assumed to be constant in the neighborhood. Equation (1) implies that we should look at each (x,y) for $\underline{v} = (v_x, v_y)$ such that

$$\int_{\mathrm{patch}_i} (\tilde{E}_t(x,y) - \tilde{E}_{t+\Delta t}(x + v_x\Delta t, y + v_y\Delta t))^2 dx dy \tag{2}$$

is minimized. Alternatively, one can maximize the negative of the integrated result. Equation (2) represents the sum of the pointwise squared differences between a patch in the first image centered around the location (x, y) and a patch in the second image centered around the location $(x + v_x\Delta t, y + v_y\Delta t)$.

This algorithm can be translated easily into the following description. Consider a network of processors representing the result of the integrand in equation (2). Assume for simplicity that this result is either 0 or 1 (this is the case if E_t and $E_{t+\Delta t}$ are binary feature maps). The processors hold the result of differencing (taking the logical "exclusive or") of the right and left image map for different values of (x, y) and v_x, v_y. The next stage, corresponding exactly to the integral operation over the patch, is for each processor to summate the total (2) in an (x, y) neighborhood at the same disparity. Each processor thus collects a vote indicating support for a patch of surface existing at that displacement. The algorithm iterates over all displacements in the range $\pm\delta, \pm\delta$, recording the values of the integral (2) for each displacement. The last stage is to choose $\underline{v}(x, y)$ from among the displacements in the allowed range that maximizes the integral. This is done by an operation of "non-maximum suppression" across velocities from the finite allowed set: at the given (x, y), the processor is found that has the maximum vote. The corresponding $\underline{v}(x, y)$ is the velocity of the surface patch found by the algorithm. The actual implementation of this scheme can be simplified so that the "non-maximum suppression" occurs during iteration over displacements, so that no actual table of summed differences over displacements need be constructed. In practice, this formalism has been shown to be effective both for synthetic and natural images using many image transformations before comparison, including edge detectors (both zero-crossings of the Laplacian of Gaussian and Canny's method) which generate sparse results along intensity edges, and direct use of intensities or the sign of the Laplacian of Gaussian, which generate dense results.

2.2.4. Texture

Texture computations, specifically, determining texture boundaries, is an important component of the Vision Machine. Such boundaries provide yet another cue to determine object boundaries, since the projection of object boundaries often coincides with texture changes. Voorhees [1987] has described several schemes for reliably determining texture boundaries in images. One simple scheme first performs filtering with the Laplacian of Gaussian, then thresholds the filter image at some non-zero value, yielding blobs which characterize the textural elements in the textured regions. To compute the density of the blobs, or the amount of area locally covered by the blobs, it is sufficient to sum locally the number of pixels which are in the blobs. This gives a useful discriminator which can identify some textures. This algorithm has a particularly simple parallel implementation on the Connection Machine, since there exist routines, relying on the *scan* capability of the Connection Machine, which can rapidly compute such local sums everywhere in an image. These computations are prohibitively expensive on a serial computer. Further enhancements of these texture discriminators involve orientation selectivity and thinning operations on the blobs derived above. We are actively working on these developments: they also have simple implementations on the Connection Machine. Other textural discrimination methods summate, for example, the energy in the image filtered by the Laplacian of Gaussian. This fits into the framework, and appears to be useful in some cases.

2.3. The MRF Integration Scheme

Integration of several vision modules is likely to be one of the main keys to the power and robustness of the human visual system. The problem of integrating early vision cues is clearly the central problem in our Vision Machine project. E. Gamble and T. Poggio have suggested that integration is best performed at the location of discontinuities in early processes, such as discontinuities in depth, intensity, motion and texture. They have used coupled Markov Random Field (MRF) models based on Bayesian estimation techniques, to combine vision modalities with their discontinuities. They have derived a scheme to integrate intensity edges with stereo depth and motion field information, and show results on synthetic and natural images. The use of intensity edges to integrate other visual cues and to help to discover discontinuities emerges as a general and powerful principle. These models generate algorithms that map naturally on parallel fine-grained architectures such as the Connection Machine. Gamble and Poggio have chosen to use the machinery of Markov Random Fields, initially suggested for image processing by Geman and Geman [1984]. Consider the prototypical problem of approximating a surface given sparse and noisy data (depth data) on a regular 2-D lattice of sites. We first define the prior probability of the class of surfaces we are interested in. The probability of a certain depth at any given site in the lattice depends only upon neighboring sites (the Markov property). Because of the Clifford-Hammersley theorem, the prior probability is guaranteed to have the Gibbs form:

$$P(f) = \frac{1}{Z} e^{-\frac{U(f)}{T}} \tag{3}$$

where Z is a normalization constant, T is called temperature, and $U(f) = \sum_C U_C(f)$ is an energy function that can be computed as the sum of local contributions from each neighborhood. The sum of the *potentials*, $U_C(X)$, is over the neighborhood's *cliques*. A clique is either a single lattice site or a set of lattice sites such that any two sites belonging to the set are neighbors of one another. Thus $U(f)$ can be considered as the sum over the possible configurations of each neighborhood [see Marroquin et.al., 1987; Gamble and Poggio, 1987]. As a simple example, when the surfaces are expected to be smooth, the prior probability can be given in terms of:

$$U_c(f) = (f_i - f_j)^2 \tag{4}$$

where i and j are neighboring sites (belonging to the same clique).

If a model of the observation process is available (i.e., a model of the noise), then one can write the conditional probability $P(g/f)$ of the sparse observation g for any given surface f. Bayes Theorem then allows one to write the posterior distribution:

$$P(f/g) = \frac{1}{Z} e^{\frac{U(f/g)}{T}} \tag{5}$$

In the simple earlier example, we have (for Gaussian noise):

$$U(f/g) = \sum_C \alpha \gamma_i (f_i - g_i)^2 + (f_i - f_j)^2$$

where $\gamma_i = 1$ only where data are available. More complicated cases can be handled in a similar manner [Gamble and Poggio, 1987].

The posterior distribution cannot be solved analytically, but sample distributions with the probability distribution of Equation (5) can be obtained using Monte Carlo techniques such as the Metropolis algorithm. These algorithms sample the space of possible surfaces according to the probability distribution $P(f/g)$, which is determined by the prior knowledge of the allowed class of surfaces, the model of noise, and the observed data. In our implementation, a highly parallel computer generates a sequence of surfaces from which, for instance, the surface corresponding to the maximum of $P(f/g)$ can be found. This corresponds to finding the global minimum of $U(f/g)$ (simulated annealing is one of the possible techniques). Other criteria can be used:

Marroquin [1985] has shown that the average surface f under the posterior distribution is often a better estimate and can be obtained more efficiently by simply finding the average value of f at each lattice site.

One of the main attractions of MRF is that the prior probability distribution can be made to embed more sophisticated assumptions about the world. Geman and Geman [1984] introduced the idea of another process, the line process, located on the dual lattice and representing explicitly the presence or absence of discontinuities that break the smoothness assumption. The associated energy then becomes:

$$U - c(f) = (f_i - f_j{}^2(1 - l_i^j) + \beta V_C(l_i^j) \tag{6}$$

where l is a binary line element between site i, j. V_C is a term that reflects the fact that certain configurations of the line process are more likely than others to occur. In our world, depth discontinuities are usually themselves continuous and non-intersecting, and are rarely isolated joints. These properties of physical discontinuities can be enforced locally by defining an appropriate set of energy values $V_c(l)$ for different configurations of the line process in the neighborhood of the site.

It is obviously possible to extend the energy function to accommodate the interaction of more processes and their discontinuities. In particular, Gamble and Poggio have extended the energy function to couple several of the early vision modules (depth, motion, texture and color) to intensity edges in the image. This is a central point in our integration scheme: intensity edges guide the computation of discontinuities in the other physical processes, thereby coupling surface depth, surface orientation, motion, texture and color each to the image intensity data and thus to each other. The reason for the role of intensity edges is clear: changes in surface properties usually produce large intensity gradients in the image. It is exactly for this reason that edge detection is so important in both artificial and biological vision.

The coupling to intensity edges is done by adding to U_C the term:

$$V(l, e) = \beta' P(1 - e_i^j) V_C(l_i^j) \tag{7}$$

with $e_i^j = 1, 0$, depending on the presence or absence of an intensity edge between site i, j. This term facilitates formation of discontinuities (that is, l_i^j) at the locations of intensity edges.

These highly parallel algorithms map quite naturally onto an architecture such as that of the Connection Machine, which consists of 16K simple 1-bit processors with local and gloƅal conn‿.ᵗⁱon capabilities. These algorithms also map onto VLSI architectures of fully analog elements (we have successfully experimented with a version of Equation (7) in which l is a continuous variable), mixed analog and digital components (such as directly suggested by the previous equations), and purely digital processors (similar to a much simplified and specialized Connection Machine).

2.4. Recognition

The problem of visual recognition is a many-faceted problem. We have worked on several of these sub-problems. In particular, we summarize here some of our initial work on the critical issues of how to *segment* an image in order to initiate the recognition process and reduce its intrinsic combinatorics, and how to use 2-D views to recognize 3-D objects.

2.4.1. Control Structures for Image Segmentation

J. Mahoney has begun to explore the problem of image segmentation from a new perspective. The separation of figure from ground is still very much a mystery, despite progress in "higher-level" matters such as interpretation of three-dimensional structure and object recognition. Scrutiny of traditional information processing decompositions of visual objects suggests that the root of the difficulty is that segmentation is commonly assumed to be implemented by a single module.

Mahoney argues that segmentation should instead be viewed as inherently implicit in a number of other processes – i.e., the basic operations of segmentation are selected and controlled by a number of other processes. He proposed an alternative system architecture that does away with the notion of segmentation as a module: figure-ground separation is seen as a complex behavior exhibited by the system as a whole, and implemented by the interaction between its modules. The modules of the system are elementary processes of visual object perception: structure interpretation, classification and application. The control of segmentation exerted by these processes has a layered organization: interpretation determines how figures are extracted, classification influences what figures are extracted, and application determines where and when the figure separation takes place. This layered organization implies that the various levels of control in separation may be studied somewhat independently.

Our recent studies of the basic segmentation operations (curve tracing, area coloring and indexing of salient locations) addressed the question of how fine-grained parallelism of the sort provided by the Connection Machine may be exploited in implementing these operations efficiently. Our study of the proposed control structure for segmentation has begun to examine the elementary object perception processes – structure interpretation, classification and application – and their interactions in detail. The Connection Machine provides large-scale parallelism and general purpose routing which supports efficient implementation of these processes.

2.4.2. Recognizing 3-D Objects from 2-D Views

We have implemented a system for recognizing objects in three-space from a single two-dimensional view, such as a photograph. D. Huttenlocher and S. Ullman have shown that three corresponding points are sufficient to recover the three-dimensional position, orientation and scale of a model with respect to an image. Thus our approach is to first *align* models with an image, using triples of corresponding model and image points, and then to compare the aligned models with the image. Unlike conventional tree search and graph matching approaches to recognition, our method is well suited to implementation on a massively parallel computer. Each alignment operation is independent, and thus all the alignments between models and an image can be solved in parallel.

Given a set of corresponding triples of model and image points, each triple is used to compute the position, orientation, and scale of all models containing that triple. This operation is performed in parallel for all triples. The aligned models are then sorted to group together instances of the same model. This takes time $O(k)$ for 2^k models. Finally, for each model, the number of similar alignments is determined. Models for which there are two or more independent alignments proposing a similar position, orientation and scale are likely to actually be in the image, and should be further verified.

2.5. Other Vision Algorithms

We have explored the implementation on the Connection Machine of several other algorithms, especially those with a more global flavor requiring extensive use of the router network as opposed to the NEWS network.

2.5.1. The Hough Transform

The Hough Transform is frequently used in image analysis to determine the existence of straight lines in an image [Ballard and Brown, 1981]. The Generalized Hough Transform, similarly, is used to determine the parameters of the position and orientation of a known object from an image. The Hough Transform computes an accumulator array of values, in which the (θ, ρ) entry records the number of pixels lying on a line with parameters (θ, ρ).

We parameterize lines by the normal angle θ, and the perpendicular distance from the origin ρ. The Hough Transform table will be stored in a matrix of processors indexed by (θ, ρ). We compute the Hough Transform in θ separate stages: each step computes the values for a particular angle $\theta(i)$. For each angle, we broadcast $cos\theta(i)$ and $sin\theta(i)$. Each processor computes the scalar product of its (x, y) address with the normal described by the broadcast pair. Each processor then knows its (θ, ρ). We count votes by *sending* a 1 from each active pixel, summing at the destination processor. If we were simply to *send* each vote to the table, there would be many collisions, with many messages arriving at one processor, especially when (θ, ρ) does in fact represent a line. This can be slow when the number of collisions is large (≥ 64).

The routing hardware incurs little penalty for having up to 32 collisions per destination. By randomizing the vote destinations, we can minimize the number of collisions. For each θ, a pixel computes a location based on *rho* and some random value, say, one computed at the start. For a fixed ρ, pixels *send* their votes to a contiguous block of processors. Block sizes are calculated to reduce the average number of collisions. The votes are summed when they arrive, using a built-in capability of the router. Then the votes are tallied as before, using a *plus-scan* operation, with segments. This will achieve the same result with less overhead for computing addresses.

2.5.2. Matching 2-D Descriptions with 3-D Rock Models

More work has been done on the matcher discussed by W. Lim [in Proceedings of the Image Understanding Workshop, 1987]. The matcher discussed in that work uses 2-D descriptions, represented by a graph termed a *model graph*, for recognizing landmarks which are objects in the rocks world. The 2-D descriptions are derived from contour segments of the occluding contour of the object. The matcher has been implemented on the Connection Machine and has been tested on objects using manually created models for three real objects. Synthetic models are also used by changing some of the attributes in the models for the real objects. The number of models in the data base is approximately one-tenth the number of nodes in the CM, since each model graph has about ten nodes as well as nodes mapped directly to processors. The sizes of the CMs used are 8K and 16K. Due to insufficient memory in the processors of the CM-1, implementation of the algorithm has been moved to the CM-2, access to which Thinking Machines Corporation has provided access.

The algorithm has been extended to match graphs that are more complete, i.e., they contain 2-D descriptions derived from the internal structure of the objects in addition to those obtained from the occluding contour. The additional information includes the number of observed surface patches that are not on the occluding contour, and possibly their surface types, e.g., flat or curved. The algorithm first finds views of models that match the occluding contour. From this set of views, it then looks for those that will also satisfy the constraints imposed by the internal structure of the object in the scene. Work is in progress on implementing the extension and testing the entire algorithm using larger data bases, perhaps as large as will fit on the 64K CM-2.

2.6. Other Architectures

Most of our work has been focused on the Connection Machine, as we had originally proposed, in order to establish the strength and utility of fine-grained parallel architectures for vision within a navigation task. Thanks to the in-house work by R. Brooks on the Mobile Robot, we have found it interesting to contrast the fine-grained architecture of the CM-1 with the subsumption architecture used by Brooks.

2.6.1. Architectures for Robot Navigation

We have examined system architectures for robotic navigation, in the context of Brooks' layered systems for building robots. Brooks' subsumption architecture for robots provides an environment for building robots which allows us to investigate several forms of exploratory behavior. In this system, we can study various forms of spatial reasoning [Connell, 1986].

Recently, we have been exploring the application of Brooks' subsumption architecture to robotic navigation. This architecture is attractive because a complete system can be built and debugged incrementally. The architecture also allows for a graceful degradation of performance if the robot encounters unusual environments, or if some of the subsystems become inoperative.

To test these ideas, we have investigated two types of exploratory behavior cast in the subsumption framework [Brooks and Connell, 1986]. We are also looking at how to build and maintain coarse maps of the environment using similar techniques [Brooks and Connell, 1987]. Finally, we are probing ways to extend the architecture to allow certain other types of spatial reasoning. In particular, we are interested in having the robot identify, pick up, and then return certain types of objects [Connell, 1986].

The conclusion we have reached is that the two architectures can be complementary for navigation. The simple subsumption architecture can underly simple reflexive behaviors of the insect type. For more sophisticated tasks involving planned visual navigation and recognition, however, the power of a parallel supercomputer is barely sufficient, given the complexity of the tasks.

2.7. Parallel Languages

*LISP is a good language for early and middle vision. For more abstract tasks, other languages would be desirable. It is questionable whether Thinking Machines Corporation will be able to develop these more abstract languages, since its main effort at present is focused on extending standard languages such as C and even FORTRAN. We have been comparing three parallel versions of Lisp that have been designed for the Connection Machine. These three languages are *LISP, Paralation Lisp and CM-Lisp. *LISP is the current parallel Lisp supported by Thinking Machines for the Connection Machine. CM-Lisp is a language designed by D. Hillis and G. Steele for the Connection Machine: it is a much higher level language. A simulator and a preliminary compiler have been implemented at Thinking Machines. Paralation Lisp is a language designed by G. Sabot, a graduate student at Harvard University. It is similar to CM-Lisp, but is not at such a high level (its model is slightly closer to the machine model). A simulator for the language has been implemented by Sabot.

The purpose of comparing the three languages is to determine what the best abstractions are for a parallel Lisp, and also to select the features of the three languages that are most important. We feel that this knowledge will help us to decide on implementation languages in the future, and will reduce our coding effort.

The general approach taken is to code the same algorithms in the three languages and to get a feeling of what helpful features are either present or lacking in each language. We also try to get a feeling of which language most clearly and concisely expresses the algorithms. In addition, we are studying compilers for the three languages so that we can determine how well a compiler can compile the algorithms into an efficient set of Connection Machine instructions.

Some subgoals attained this summer include the implementation of several algorithms in Paralation Lisp, for example:

A sorting algorithm (Quicksort)

A convex hull algorithm (Quickhull)

A line drawing algorithm

A K-D tree algorithm

An algorithm to determine the Shannon entropy of a sequence of symbols

An algorithm to build Quinlan Decision trees.

We have also sped up the simulator for Paralation Lisp by a factor of 100, and implemented a preliminary compiler for Paralation Lisp. In addition, we have implemented some algorithms in CM-Lisp including:

A sorting algorithm (Quicksort)

A line drawing algorithm

An algorithm to determine the Shannon entropy of a sequence of symbols.

3. RELEVANT TECHNICAL REPORTS AND ABSTRACTS

1. Agarwal, A., L. Nekludova and W. Lim. "A Parallel $o(\log n)$ Algorithm for Finding Connected Components in Planar Images," *Proc. Intl. Conf. on Parallel Processing*, 783-786, August, 1987.

2. Blelloch, G. "Scans as Primitive Parallel Operations," *Proc. Intl. Conf. on Parallel Processing*, 355-362, August, 1987.

3. Blelloch, G. and J. Little. "Parallel Solutions to Geometric Problems on the Scan Model of Computation," *Artificial Intelligence Laboratory Memo 952*, Massachusetts Institute of Technology , Cambridge, MA, 1987.

4. Blelloch, G. and C. Rosenberg. "Network Learning on the Connection Machine," *Proc. Intl. Joint. Conf. on Artificial Intell.*, 323-326, August, 1987.

5. Brooks, R. and J. Connell. "Asynchronous Distributed Control Systems for a Mobile Robot," *Proc. S.P.I.E.*, no. 727, October, 1986.

6. Brooks, R. and J. Connell. "Navigation Without Representation," *Artificial Intelligence Laboratory Technical Report*, Massachusetts Institute of Technology , Cambridge, MA, in progress.

7. Connell, J. "Task Oriented Spatial Representation for Distributed Systems," *Artificial Intelligence Laboratory Technical Report*, Massachusetts Institute of Technology , Cambridge, MA, September, 1986.

8. Gamble, E. and T. Poggio. "Integration of Intensity Edges with Stereo and Motion," *Artificial Intelligence Laboratory Memo 970*, Massachusetts Institute of Technology , Cambridge, MA, October, 1987.

9. Hillis, W.D. **The Connection Machine**, The MIT Press, Cambridge, MA, 1985.

10. Huttenlocher, D. and S. Ullman. "Object Recognition Using Alignment," *Proc. Intl. Conf. on Computer Vision*, 102-111, June, 1987.

11. Lim, W. "Fast Algorithms for Labelling Connected Components in 2-D Arrays," *TMC Technical Report NA86-1*, Thinking Machines Corp., Cambridge, MA, December, 1986.

12. Lim, W. "Using Occluding Contours for Object Recogition," *Proc. Image Understanding Workshop*, DARPA, 909-914, February, 1987.

13. Lim, W., A. Agarwal and L. Nekludova. "A Fast Parallel Algorithm for Labeling Connected Components," *TMC Technical Report NA86-2*, Thinking Machines Corp., Cambridge, MA, December, 1986.

14. Little, J., G. Blelloch and T. Cass. "How to Program the Connection Machine for Computer Vision," *Proc. Workshop on Comp. Arch. for Pattern Anal. and Mach. Intell.*, October, 1987.

15. Little, J., G. Blelloch and T. Cass. "Parallel Algorithms for Computer Vision on the Connection Machine," *Proc. Image Understanding Workshop*, DARPA, 628-638, February, 1987.

16. Little, J., G. Blelloch and T. Cass. "Parallel Algorithms for Computer Vision on the Connection Machine," *Proc. Intl. Conf. on Computer Vision*, 587-591, June, 1987.

17. Little, J., H. Bülthoff and T. Poggio. "Parallel Optical Flow Computation, *Proc. Image Understanding Workshop*, DARPA, 915-920, February, 1987.

18. Mahoney, J. "Image Chunking: Defining Spatial Building Blocks for Scene Analysis," Master's Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology , Cambridge, MA, 1986.

19. Poggio, T. and the Staff of the A.I. Laboratory. "MIT Progress in Understanding Images," *Proc. Image Understanding Workshop*, DARPA, 41-54, February, 1987.

20. Verri, A. and T. Poggio. "Against Quantitative Optical Flow," *Proc. Intl. Conf. on Computer Vision*, 171-180, June, 1987.

21. Verri, A. and T. Poggio. "Qualitative Information in the Optical Flow," *Proc. Image Understanding Workshop*, DARPA, 825-834, February, 1987.

22. Voorhees, H. and T. Poggio. "Detecting Textons and Texture Boundaries in Natural Images," *Proc. Intl. Conf. on Computer Vision*, 250-258, June, 1987.

23. Voorhees, H. and T. Poggio. "Detecting Blobs as Textons in Natural Images," *Proc. Image Understanding Workshop*, DARPA, 892-899, February, 1987.